C. A. Conroy

International Business Machines Corporation

Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598
914/945–3000

APL\360 *NEWSLETTER NO.* 2

DECEMBER, 1967

On October 25, *APL*\360 celebrated its first anniversary. In the first year of operation, there were more than 30,000 terminal hours of use and we now have over 900 users on the system.

The normal *APL*\360 schedule is 10:00 a.m. to 9:00 p.m. EST. But, when possible, it is kept up until 10:00 p.m. or later.

When sending messages to the operator, if a reply is expected, use the system command)*OPR*. Please then leave your terminal in the locked state and wait for the operator's reply. If no reply is required, the system command)*OPRN* can be used.

APL SYSTEM LIBRARY

Included in this issue are two new system library workspace descriptions and a reissued description for *CLEANSPACE* in Library 1. The two new workspace descriptions are:

WORKSPACE NAME	AUTHOR
APLCOURSE	A. J. Rose
TYPE	A. J. Rose

The description of *CLEANSPACE* is being reissued to amplify the earlier information and also take into account the recently implemented workspace save protective feature. The operation of this feature, which prevents a user from inadvertently destroying a workspace by saving information into a workspace which has not been loaded, is given in the section on new features.

PROGRAMMING HINTS

The following is an actual sequence of debugging a function while executing it for the first time. It demonstrates the use of the line-editing facility for making insertions, deletions, and changes in a program statement.

Note that by repeatedly asking for continuation, (by →1), only one execution of the function was necessary. It is good practice to use the continuation feature, whenever possible, since it saves time and avoids cluttering the workspace with the debris of partial function executions. Where continuation is not logically feasible, branch to zero (→0) before calling the function again.

```
      ∇GUESS2[⎕]∇
    ∇ R←GUESS2 A;X;Y
[1]   R←Z[(+/(A∘.≥A-(X>∧A∘.=A)ιX←ιρA]
    ∇
      R←GUESS2 TRY                               TRY is a vector of numbers.
SYNTAX ERROR
GUESS2[1] R←Z[(+/(A∘.≥A-(X>∧A∘.=A)ιX←ιρA]
                           ∧
      ∇GUESS2[1⎕10]
[1]   R←Z[(+/(A∘.≥A-(X>∧A∘.=A)ιX←ιρA]             Editing Mode
                       2                         Two spaces required
[1]   R←Z[(+/(A∘.≥A-(X>X)∧A∘.=A)ιX←ιρA]           Insertion of missing
[2]   ∇                                             variable, parenthesis
      →1                                         Branch to Line 1
RANK ERROR
GUESS2[1] R←Z[(+/(A∘.≥A-(X>X)∧A∘.=A)ιX←ιρA]
                         ∧
      ∇GUESS2[1⎕20]
[1]   R←Z[(+/(A∘.≥A-(X>X)∧A∘.=A)ιX←ιρA]           Editing Mode
                      2                          Two spaces required
[1]   R←Z[(+/(A∘.≥A-(X∘.>X)∧A∘.=A)ιX←ιρA]         Insertion of missing
[2]   ∇                                             operators
      →1                                         Branch to Line 1
RANK ERROR
GUESS2[1] R←Z[(+/(A∘.≥A-(X∘.>X)∧A∘.=A)ιX←ιρA]
                       ∧
      ∇GUESS2[1⎕24]
[1]   R←Z[(+/(A∘.≥A-(X∘.>X)∧A∘.=A)ιX←ιρA]         Editing Mode
                   1                             One space required
[1]   R←Z[(+/(A∘.≥A)-(X∘.>X)∧A∘.=A)ιX←ιρA]        Insertion of missing
[2]   ∇                                             parenthesis
      →1                                         Branch to Line 1
VALUE ERROR
GUESS2[1] R←Z[(+/(A∘.≥A)-(X∘.>X)∧A∘.=A)ιX←ιρA]
            ∧
      ∇GUESS2[1⎕15]
[1]   R←Z[(+/(A∘.≥A)-(X∘.>X)∧A∘.=A)ιX←ιρA]        Editing Mode
        /1                                       One deletion, one space
[1]   R←A[(+/(A∘.≥A)-(X∘.>X)∧A∘.=A)ιX←ιρA]        Insertion of A which
[2]   ∇                                             replaces Z
      →1                                         Branch to Line 1
```

PROGRAMMING HINTS (Cont.)

Execution was completed although the result is not shown here. Here is the final function definition for you to analyze:

```
      ∇ R←GUESS2 A;X;Y
[1]     R←A[(+/(A∘.≥A)-(X∘.>X)∧A∘.=A)⍳X←⍳⍴A]
      ∇
```

It is also a good practice to regularly use the system command)Λ during a debugging session, to be sure there are no suspended functions. This command displays the state indicator, which lists all functions whose execution has not been completed. An *S* follows those which are suspended. Any function listed without an *S* cannot be edited until its status is changed. The entire stack of unfinished executions can be eliminated in two ways:

1. Branch to zero as many times as there are lines indicated by)Δ. This does not always work and is best used when the state indicator is relatively short.

2. Copy the workspace into a clean space, as detailed in the description of 1 *CLEANSPACE*.

It is almost always the case that 'mysterious' behavior of variables is due to the confusion between local and global variables of the same name, which occurs when functions with local variables are suspended. The procedures given here will usually clear this up.

* * *

The last newsletter contained two functions to be analyzed - *FOODFORTHOUGHT* and *GUESS*. They showed how the representation and I-beam functions (⌶) can be used. As you have probably guessed:

FOODFORTHOUGHT gives the date, formatted with slashes between the numbers.

GUESS gives the time of day. The left argument X indicates in which time system you desire to have the time of day. The right argument Y is a scalar which indicates whether you desire hours, minutes, seconds or 60th of seconds in a left to right sequence. If Y is 1, hours only would be given. If Y is 2, hours and minutes would be given, etc.

Following the display of the functions below, there are a few examples of actual executions of these functions:

```
        ∇FOODFORTHOUGHT[⎕]∇
      ∇ R←FOODFORTHOUGHT
[1]     R←1+(6⍴10)⊤⌶25
[2]     R←'/0123456789'[1+ 1 1 0 1 1 0 1 1 \R]
      ∇
        ∇GUESS[⎕]∇
      ∇ R←X GUESS Y
[1]     R←(4αY)/(X, 60 60 60)⊤⌶20
      ∇

        R←FOODFORTHOUGHT
        R
01/04/68

        FOODFORTHOUGHT
01/04/68

        TIME←24 GUESS 3
        TIME
16  34  34

        24 GUESS 2
16  35

        12 GUESS 3
4  35  21
```

APL PUZZLES

1. Find an expression for a two-element vector which will result in the first element consisting of the integer portion and the second element consisting of the

APL PUZZLES (Cont.)

fractional portion of a positive number.

2. Determine the result of the following expression: (x is a scalar)

   `(X÷⌽⍳N)⊥1`

These puzzles were contributed by L. M. Breed.

3. You were asked in the last newsletter to prove the equivalence between

   1. `M×⌈N÷M` and

   2. `N+M|-N`.

   Starting with 2., and using the relation

   `(A|B)≡A×1|B÷A`, we get

   3. `N+M×1|-N÷M`

   Factoring an *M* from this, we get

   4. `M×(N÷M)+1|-N÷M`.

   However, it is generally true that

   `(A+1|-A)≡⌈A`

   From which follows:

   5. `M×⌈N÷M`

NEWLY IMPLEMENTED *APL* FEATURES

Since the last newsletter, many *APL* features have been implemented. They include:

1. Monadic and dyadic transpose.
2. Change of dieresized characters to underlined characters.
3. Red/black ribbon shift for the 1050 terminals.
4. Trace.
5. Stop.

NEWLY IMPLEMENTED *APL* FEATURES (Cont.)

6. Natural Logarithm.
7. Reversal.
8. Protective save feature.
9. Sign-on with password.
10. Use of *CONTINUE* workspace.
11. Function display from line N.
12. Use of system command *CONTINUE*.

A description of the above features can be obtained by)*LOAD* 1 *NEWS* and execution of *APLNOW* 9 12 67. More detailed information on some of these features will be given in future issues of this newsletter . In this issue, we are including a table that shows the behavior of the transposition operator.

MISCELLANEOUS

We are also including two *APL* keyboard stickers. ✓✓

NOTE: Portions of this newsletter were prepared by using the *APL* text editing programs located in Library 10 under the workspace called *EDIT*. The text editing programs were developed by M. Zyrl and A. Mullery.

	ρ*R*	Definition
R←1 ⍉ *V*	ρ*V*	*R*←*V*
R←1 2 ⍉ *M*	ρ*M*	*R*←*M*
R←2 1 ⍉ *M*	(ρ*M*)[2 1]	*R*[*I*;*J*]←*M*[*J*;*I*]
R←1 1 ⍉ *M*	⌊/ρ*M*	*R*[*I*]←*M*[*I*;*I*]
R←1 2 3 ⍉ *T*	ρ*T*	*R*←*T*
R←1 3 2 ⍉ *T*	(ρ*T*)[1 3 2]	*R*[*I*;*J*;*K*]←*T*[*I*;*K*;*J*]
R←2 3 1 ⍉ *T*	(ρ*T*)[3 1 2]	*R*[*I*;*J*;*K*]←*T*[*K*;*I*;*J*]
R←3 1 2 ⍉ *T*	(ρ*T*)[2 3 1]	*R*[*I*;*J*;*K*]←*T*[*J*;*K*;*I*]
R←1 1 2 ⍉ *T*	(⌊/(ρ*T*)[1 2]),(ρ*T*)[3]	*R*[*I*;*J*]←*T*[*I*;*I*;*J*]
R←1 2 1 ⍉ *T*	(⌊/(ρ*T*)[1 3]),(ρ*T*)[2]	*R*[*I*;*J*]←*T*[*I*;*J*;*I*]
R←2 1 1 ⍉ *T*	(⌊/(ρ*T*)[2 3]),(ρ*T*)[1]	*R*[*I*;*J*]←*T*[*J*;*I*;*I*]
R←1 1 1 ⍉ *T*	⌊/ρ*T*	*R*[*I*]←*T*[*I*;*I*;*I*]

TABLE OF TRANSPOSITION EXAMPLES

IN GENERAL, (ρ*R*)[*I*]≡⌊/(*N*=*I*)/ρ*T*

WORKSPACE DESCRIPTION

Library: 1
Name: *APLCOURSE*

The main functions of this library workspace are:

TEACH
EASYDRILL

All other functions in this workspace are used as subfunctions with the main function. They are not self-contained.

SYNTAX	DESCRIPTION
TEACH	A flash card exercise in *APL* functions using scalars and vectors. The function prints out the choices and options available to the student. Examples are selected at random with a random starting point, and three trials are allowed for each question.
EASYDRILL	This function is the same as *TEACH* except that the problems selected are generally simpler in structure. In particular, problems involving vectors of length zero or one are excluded.

NOTE: For either function, a response of *PLEASE* will disclose the proper answer. A response of *STOP* will terminate the drill.

INDEX ORIGIN:	1
WIDTH:	120
AUTHOR:	A. J. Rose, IBM Corporation, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598

DECEMBER, 1967

APL\360 C. A. Conroy, Ed.

International Business Machines Corporation

Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598
914/WG 5–3000

WORKSPACE DESCRIPTION

Library: 1
Name: *CLEANSPACE*

The availability of *CLEANSPACE* provides a means for making a fresh start during the course of a working session. Execution of *)LOAD* 1 *CLEANSPACE* replaces the active workspace with one that is completely clean. If it is necessary to keep the information currently in the active workspace it must be saved prior to loading *CLEANSPACE*.

Common situations calling for the use of *CLEANSPACE*, and the appropriate sequence of operations are given below.

To start a new job, or provide a clean slate for assembling functions and variables from workspaces already on file in libraries, execute:

```
)LOAD 1 CLEANSPACE
```

and proceed with the appropriate activity.

In all other situations, where recovery is not possible by simply reloading a copy of the active workspace from the library, execute:

```
)SAVE        WSNAME
)LOAD   1    CLEANSPACE
)COPY        WSNAME
)DROP        WSNAME
)SAVE        WSNAME
```

If a permanent library copy is not desired, the last *)SAVE* can be omitted, or a shorter sequence using the *CONTINUE* workspace may be used:

```
)SAVE        CONTINUE
)LOAD   1    CLEANSPACE
)COPY        CONTINUE
```

(In this case there is the slight risk that if the telephone connection is broken between the second and third steps, *CONTINUE* will automatically save a copy of *CLEANSPACE*).

One notable effect of these procedures is that the state indicator - displayed by)Δ - is cleared.

Situations calling for the use of the *SAVE-LOAD-COPY* sequence are:

1. To recover from a deeply nested sequence of suspended functions, as may result from bugs in a recursive arrangement (evidenced by *DEPTH ERROR*), or simply from an accumulation of partially executed functions

(evidenced by inability to edit a function). The depth of suspension is measured by ρ⌶27 , and the functions involved can be seen by executing)Λ.

2. To recover from a loop involving ⎕ or ⍞. It is possible to execute system commands in a ⎕-condition, and one of the sequences above can be executed. In the case of ⍞, the only recovery is to break the telephone connection, which will cause the automatic execution of the equivalent of)*SAVE CONTINUE*. Then, sign on again and proceed with)*COPY CONTINUE*.

3. To maximize the available space in a workspace. Certain operations leave internal debris which, while it does not interfere with normal activities, nevertheless uses up space which would otherwise be available. Copying a workspace deletes the debris.

FUNCTIONS:	NONE
VARIABLES:	NONE
ORIGIN:	1
WIDTH:	130

DECEMBER, 1967

APL\360 C. A. Conroy, Ed.

International Business Machines Corporation

Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598
914/WG 5–3000

WORKSPACE DESCRIPTION

Library: 1
Name: *TYPE*

MAIN FUNCTION: TYPEDRILL

All other functions in this workspace are used as subfunctions.

SYNTAX DESCRIPTION

TYPEDRILL A timed typing exercise. There are three basic states which can exist: CONTROL state, ENTRY state and DRILL state.

When in the CONTROL state, typing ENTER causes the entry state to occur. In this state, you type one-line sentences, characters or APL expressions. Each typed line then represents a sentence or expression in which you wish to be drilled. TYPING ONLY A CARRIAGE RETURN CAUSES A RETURN TO THE CONTROL STATE.

Typing DRILL when in the CONTROL state will cause the DRILL state to occur. In this state the lines you entered in the ENTER state are randomly produced one at a time. When a correct entry has been made a different line will appear, until each line has been presented once. The random cycle then begins again. If the line is error-free, the time required to type the line is produced. AGAIN, TYPING ONLY A CARRIAGE RETURN CAUSES A RETURN TO THE CONTROL STATE.

Typing STAT when in the CONTROL state causes the accumulated statistics to be displayed. The vertical axis of the display is time in seconds and number of errors and the horizontal axis is trial numbers. The symbol for time is the star, the symbol for error is the circle. A time which exceeds the limit of the graph is denoted by a vertical arrow.

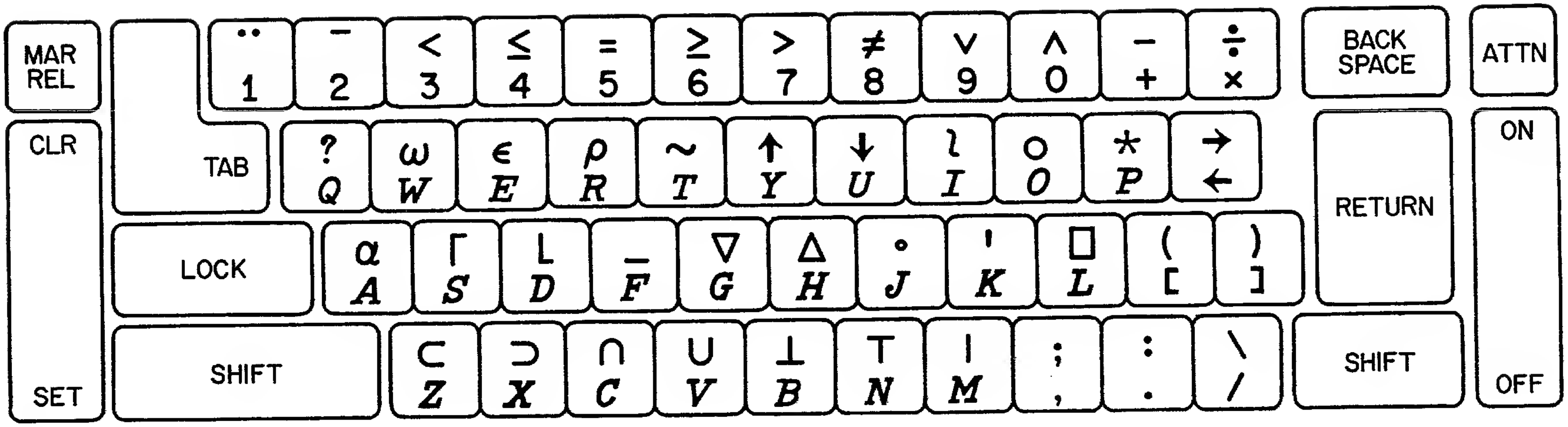
In the example display of statistics shown on the following page, there were two exercises: '*NOW IS THE TIME FOR ALL GOOD MEN*' and '*THE QUICK BROWN FOX*'. For the first of these, response time was ten seconds for the first two trials, was greater than ten seconds for the third trial, and was nine seconds for the fourth trial. There were respectively two, two, and three errors on the first three trials before a correct response was given. The summary represents number of characters per second averaged over both exercises for each cycle.

Typing STOP when in the CONTROL state causes the accumulated statistics to be displayed and terminates the exercise.

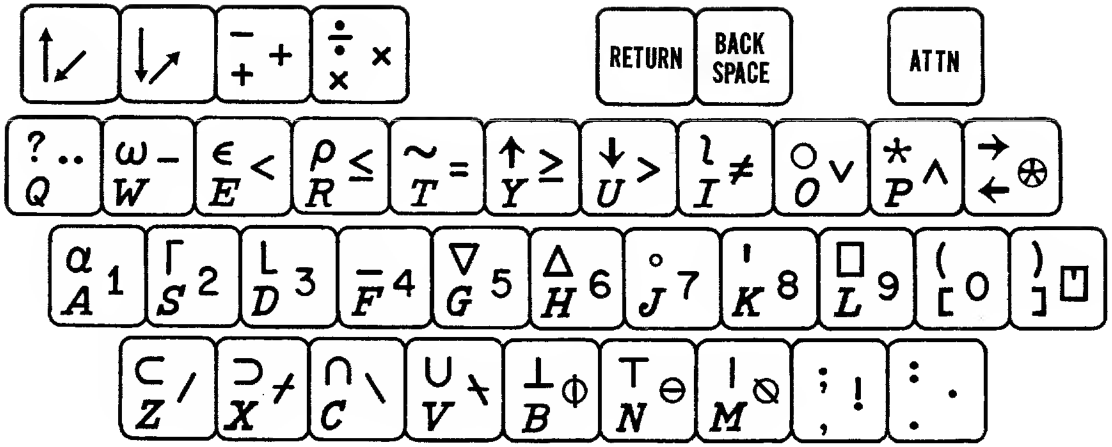
```
      STATISTICS
TIME IN SECONDS (*) AND NUMBER OF ERRORS (o) VS. TRIALS


'NOW IS THE TIME FOR ALL GOOD MEN'

10 |**↑
 9 |   **
 8 |     *  *
 7 |      ** * *
 6 |          * *
 5 |             *    *     *
 4 |              **** ***** ***
 3 |  o   oo o o
 2 |oo        o o
 1 |   oo        o    o     o
   L____________________________
TRIAL:  5   10   15   20   25

'THE QUICK BROWN FOX'

 9 |* ↑
 8 | *
 7 |   *
 6 |    **  **
 5 |           **
 4 |      **  *  *    *   *  *
 3 |              **** *** ** **
 2 | o         oo
 1 |  oo  oo  o  o    o   o  o
   L____________________________
TRIAL:  5   10   15   20   25


SUMMARY--CHARACTERS PER SECOND VS TRIALS

 6 |             *↑↑↑↑*↑↑↑*↑**↑↑
 5 |      **  * *
 4 |     *  ** *
 3 |** **
 2 |  *
 1 |
   L____________________________
TRIAL:  5   10   15   20   25
```

INDEX ORIGIN: 1
WIDTH: 120
DECEMBER, 1967

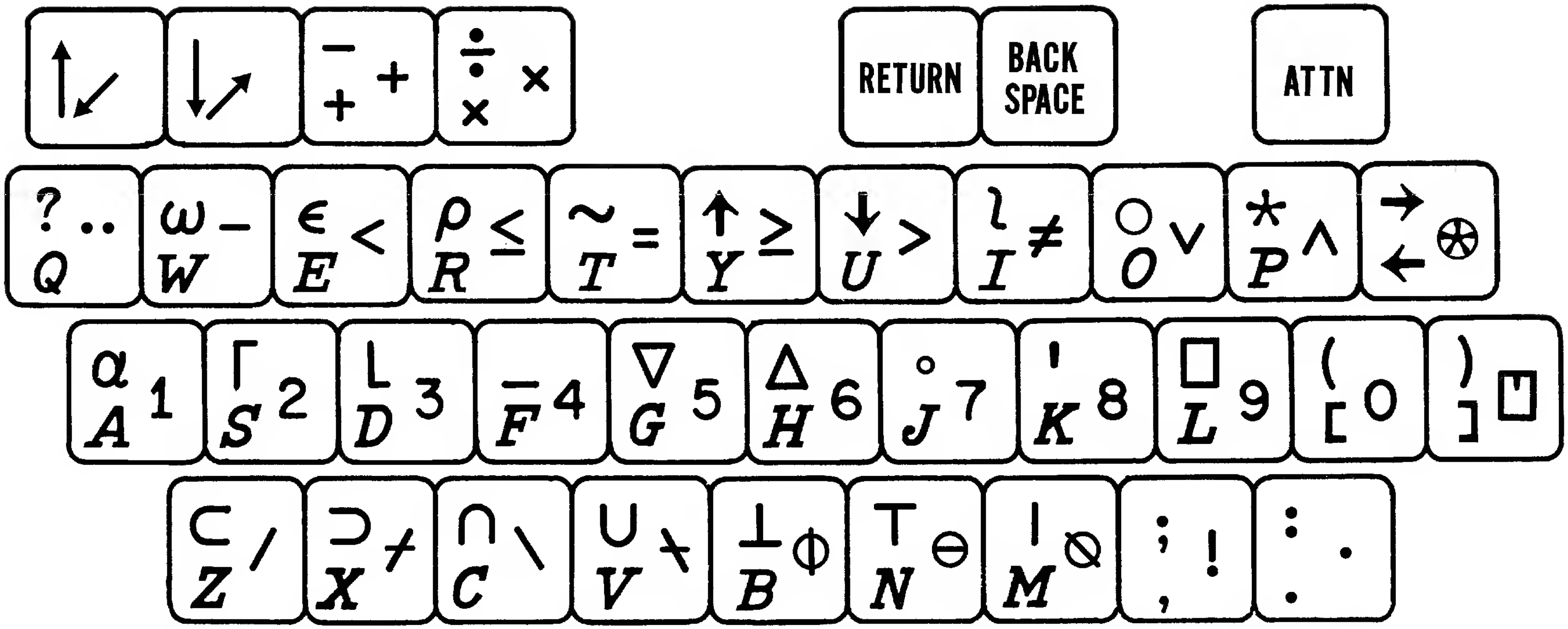
AUTHOR: A. J. Rose, IBM Corporation, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598



APL\360 KEYBOARD



RETURN
BACK SPACE
ATTN
? .. Q
ω _ W
∊ < E
ρ ≤ R
~ = T
↑ ≥ Y
↓ > U
⍳ ≠ I
○ ∨ O
⋆ ∧ P
→ ← ⍟
α 1 A
⌈ 2 S
⌊ 3 D
_ 4 F
∇ 5 G
∆ 6 H
∘ 7 J
' 8 K
⎕ 9 L
(0 [
) ⌷]
⊂ / Z
⊃ ⌿ X
∩ \ C
∪ ⍀ V
⊥ ⌽ B
⊤ ⊖ N
| ⍉ M
; ! ,
: . .



↑ ↙
↓ ↗
− + +
÷ × ×
RETURN
BACK SPACE
ATTN
? Q ..
ω W _
∊ E <
ρ R ≤
~ T =
↑ Y ≥
↓ U >
⍳ I ≠
○ O ∨
* P ∧
→ ← ⍟
α A 1
⌈ S 2
⌊ D 3
¯ F 4
∇ G 5
∆ H 6
∘ J 7
' K 8
⎕ L 9
([0
)] ⍞
⊂ Z /
⊃ X ⌿
∩ C \
∪ V ⍀
⊥ B ⌽
⊤ N ⊖
| M ⍉
; ; !
: . .